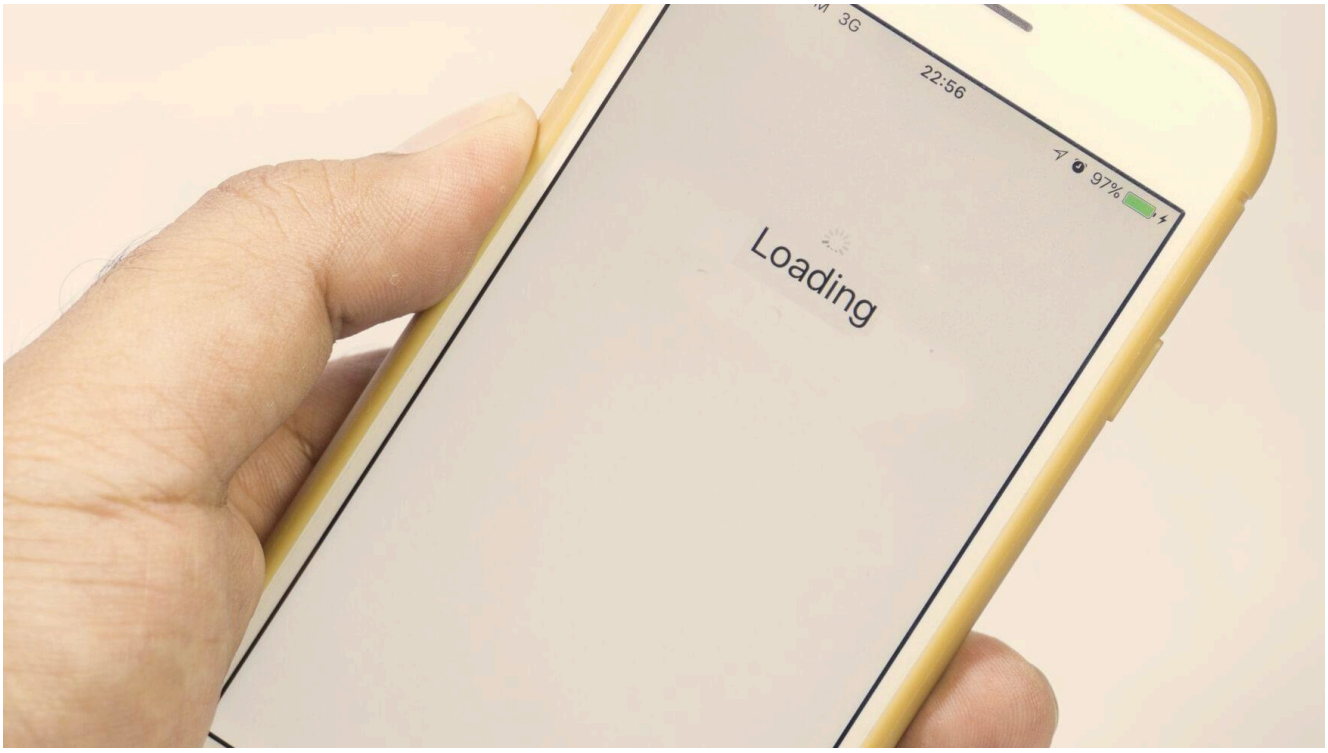
Splash Screen in Android: impariamo a crearle nel modo corretto



Introduzione

Per molte persone potrebbe sembrare un tema semplice o di cui in realtà non ci sia molto da investire in termini di studio ma spesso in realtà non è così facile creare una Splash Screen in Android.

Recentemente mi è capitato, per un progetto personale, di dover realizzarne una e ho voluto documentarmi a fondo sul modo corretto di farlo e ho deciso di scrivere questo articolo per chi come me deve affrontare questa esigenza e intenda sapere come gestirla al meglio.

Perchè?

Perchè dovrei realizzare una Splash Screen? La domanda in questo caso penso sia quasi ovvia e i motivi principali sono:

- Non è di sicuro bello esteticamente avviare un’app e stare davanti a un’activity che rimane bianca finchè i dati non vengono caricati.
- Bisogna intrattenere l’utente in modo che l’attesa sia meno pesante per non rischiare di perderlo per un motivo così banale.
- Riuscire a “riempire” lunghi tempi di attesa se la nostra app deve caricare una considerevole quantità di dati iniziali.

A favore o contro?

Se dovessi schierarmi lo farei nettamente a favore. È innegabile il fatto che la nostra applicazione presenterà un lasso di tempo più o meno consistente per visualizzarsi (creazione dei controlli e caricamento dati sono i fattori principali). Da qui lo spunto per sfruttare questo tempo per creare un prodotto che si presenti più professionale senza dover infastidire l’utente.

Questa ultima frase rappresenta appunto il nocciolo di questo articolo. Creare una Splash Screen può essere molto banale in realtà, ma quanti sanno qual’è il modo migliore per farlo?

Analizzeremo i 3 modi più diffusi partendo da quello peggiore (ma molto utilizzato), a quello intermedio fino a guingere a quello migliore.

definizione e scopo

Innanzitutto partiamo dalla definizione che ci viene data dalle linee guida del Material Design:

> *Launch screens can be displayed upon an app’s launch from the home screen when an app loads, instead of displaying a blank screen. Displaying a launch screen can decrease the sense of a long load time, and has the potential to add delight to the user experience.*

La prima cosa che possiamo notare è che il termine corretto è **Launch Screen** e d'ora in poi in questo articolo utilizzerò questo termine. Nel titolo dell'articolo è stato utilizzato apposta "Splash Screen" perchè in realtà è il termine più utilizzato anche se improprio.

Possiamo raggruppare le Launch Screen in 2 diversi gruppi:

- Placeholder UI: viene visualizzato un layout vuoto in cui le sue varie parti vengono caricate man mano che i dati arrivano. Un esempio può essere quello seguente dove le notizie vengono inserite nei rispettivi segnaposto non appena vengono caricate.
- Branded launch: In questo caso solitamente la Launch Screen si presenta molto più statica dato che il compito principale è solo quello di visualizzare in modo chiaro il brand con il logo e/o il nome dell'app o servizio

Per approfondire meglio quelle che sono le linee guida delle 2 soluzioni vi rimando alla pagina ufficiale di material.io.

Abbiamo precedentemente accennato ai 3 metodi principali di realizzazione di una Launch Screen che andremo ad analizzare cioè:

- Utilizzando un'activity dedicata e un Timer (peggiore)
- Utilizzando un'activity dedicata (intermedio)
- Utilizzando un Launcher Theme (ottimo)

Vediamo ora di approfondire in dettaglio queste 3 diverse implementazioni.

Utilizzo di un timer

Questo caso mi fa sorridere molto perchè immagino un timer che va dai 3 ai 5 secondi in cui non si esegue nessuna riga di codice, che il povero sviluppatore ha posizionato per visualizzare una Launch Screen per dare un tocco più professionale alla sua applicazione. Prima di tutto ci tengo a precisare che anche io sono appartenuto a questa tipologia e

che soprattutto chi si approccia alla programmazione di un’app può inizialmente trovare come la strada più sbrigativa per realizzare quanto gli serve.

Realizzazione:

1. Creiamo una nuova Activity.

   ```
   import android.os.Bundle
   import androidx.appcompat.app.AppCompatActivity class LaunchScreenActivity: AppCompatActivity() { override fun onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) setContentView(R.layout.activity_launch_screen) } }
   ```

2. Nella onCreate, dopo un’attesa di X secondi, dovremo lanciare l’activity principale della nostra applicazione e chiudere la launch activity.

   ```
   Handler().postDelayed( { // Terminati i 3 secondi di attesa verrà lanciata l'activity // principale e terminata quella di launch startActivity(Intent(this, MainActivity::class.java)) finish() }, 3000 // La Launch Screen rimarrà visibile per 3 secondi
   )
   ```

3. Nel Manifest.xml dovremo indicare che l’activity da lanciare all’apertura dell’applicazione è quella di Launch Screen.

   ```
   <activity android:name=".LaunchScreenActivity" android:theme="@style/AppTheme"> <intent-filter> <action android:name="android.intent.action.MAIN" /> <category android:name="android.intent.category.LAUNCHER" /> </intent-filter>
   </activity>
   ```

La realizzazione come si può notare è molto semplice. Una

variante spesso utilizzata è quella di variare il tempo della splash se si stà lanciando l'applicazione la prima volta (quindi potrebbe aumentare il tempo perchè nella nostra classe Application dobbiamo inizializzare alcune librerie per la prima volta come ORM, injection, ecc..) e nelle successive. In questo caso la prima volta si aumenta il tempo e per le successive lo si diminuisce. Per fare questo è possibile, ad esempio, salvare nelle shared preference il primo start eseguito.

Pro:

- Essendo un'activity vera e propria sarà possibile inserire animazioni, view customizzate e tutto quello che ci viene in mente per renderla molto accattivante
- Al termine del timer è possibile lanciare qualsiasi activity in base a determinate condizioni. Ad esempio lanciare il login se non è configurato un utente, lanciare la main activity, lanciare l'activity di un evento ad una determinata data, ecc…

Contro:

- Creazione di delay non proporzionato a quello che realmente servirebbe all'inizializzazione (potrebbe risultare troppo lungo o addirittura troppo corto se il nostro dispositivo risultasse già quasi saturo di memoria o impegnato in altre operazioni costose)
- Più animazioni o view sono presenti più aumenta al lancio la visualizzazione del solo background dell'activity perchè deve ancora essere caricato tutto il contenuto.
- Una Launch Screen di questo genere può risultare molto accattivante ma non bisogna sottovalutare che un utente si può anche stancare della troppa attesa e quindi in atto estremo preferire un'altra app.

Utilizzo di un’activity dedicata

Questo scenario è molto simile a quello descritto precedentemente ma ne rappresenta un miglioramento. Consiste nel creare un’activity dedicata per la Launch Screen e al termine delle operazioni di inizializzazione lanciare subito quella successiva.

```
import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity class LaunchScreenActivity: AppCompatActivity() { override fun onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) setContentView(R.layout.activity_launch_screen) initData() startActivity(Intent(this, MainActivity::class.java)) finish() } private fun initData() { // Inizializzazione dati utilizzati dall'applicazione }
}
```

In questo caso la durata della Launch Screen è pressochè uguale alla durata del metodo initData() nel quale potremo caricate tutti i dati che potranno essere utilizzati globalmente all’interno dell’applicazione.

Pro:

- Tutti quelli descritti nell’utilizzo del timer
- La durata dell’activity sarà quella effetivamente necassaria a svolgere le operazioni di inizializzazione

Contro:

- Operazioni troppo veloci di inizializzazione fanno si che il tempo di transizione tra la Launch Screen e la successiva activity sia troppo veloce non riuscendo neanche a completare una banale animazione.
- Come nel caso del timer, aumentando la complessità della Launch Screen aumenta la probabilità di visualizzare inizialmente solo il background dell’activity e vedere in seguito il caricamento del layout.

Utilizzo di un Launcher Theme

Questo metodo è quello che in assoluto preferisco e che cerco sempre di utilizzare se dominio me lo consente.

Come si realizza? Il concetto è quello di creare un tema da applicare subito all'activity che inizialmente verrà lanciata ancora prima di applicare il layout con setContentView e di applicare quello "reale" in seguito. Vediamo quanto spiegato con un esempio pratico.

Definiamo innanzitutto nel file styles.xml il tema di launch sreen in questo modo:

```
<style name="AppTheme.Launcher"> <item name="android:windowBackground">@drawable/launcher_screen</item> <item name="android:windowNoTitle">true</item> <item name="android:windowActionBar">false</item> <item name="android:windowFullscreen">true</item> <item name="android:windowContentOverlay">@null</item> </style>
```

Come si può notare la parte più importante è rappresentata dalla riga numero 2 nella quale andiamo a definire il background con la risorsa launcher_screen presente in res/drawable.

Per realizzarla basterà creare il file launcher_screen.xml e un esempio del contenuto può essere il seguente

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list
xmlns:android="http://schemas.android.com/apk/res/android" android:opacity="opaque"> <item android:drawable="@android:color/holo_green_dark"/> <item android:bottom="150dp" android:gravity="center" android:drawable="@drawable/logo_splash"/> <item android:top="150dp" > <bitmap android:gravity="center" android:src="@drawable/app_title"/> </item> </layer-list>
```

Prestiamo particolare attenzione alle righe:

4. Viene definito il colore di background del tema

6. Rispetto al centro (android:gravity=”center”) andremo a visualizzare il logo della nostra applicazione con un margine inferiore di 150dp

10. Rispetto al centro (android:gravity=”center”) andremo a visualizzare il nome della nostra applicazione con un margine superiore di 150dp

Definito il tema dovremo applicarlo all’activity dal Manifest.xml in questo modo

```
<activity android:name=".MainActivity" android:theme="@style/AppTheme.Launcher"> <intent-filter> <action android:name="android.intent.action.MAIN" /> <category android:name="android.intent.category.LAUNCHER" /> </intent-filter>
</activity>
```

Rispetto agli esempi precedenti notiamo che in activity:name non c’è più un’activity dedicata ma quella che verrà lanciata all’avvio dell’applicazione. In activity:theme andremo a indicare il tema precedentemente creato per la launch screen.

A questo punto non ci rimane che nell’activity che viene lanciata, MainActivity nel nostro esempio, impostare il tema standard dopo l’inizializzaizone dei dati

```
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity class MainActivity: AppCompatActivity() { override fun onCreate(savedInstanceState: Bundle?) { initData() setTheme(R.style.AppTheme) super.onCreate(savedInstanceState) setContentView(R.layout.activity_launch_screen) } private fun initData() { // Inizializzazione dati utilizzati dall'applicazione }
}
```

Pro:

- Non serve creare nessuna nuova activity per la launch

screen
- Non serve impostare alcun delay ma il tema di launch screen verrà visualizzato per il solo tempo necessario all'inizializzazione dei dati
- Il caricamento del layout è immediato senza il fastidio di vedere un eventuale background caricato e successivamente i componenti creati

Contro:

- Non è possibile avere la complessità offerta dai primi 2 metodi ma solo quello che ci offre il file xml che utilizzeremo come background

Conclusioni

Sebbene il metodo che preferisco sia quello di usare un tema ad hoc per la launch screen perchè la cosa più importante che salta all'occhio è la visualizzazione immediata del contenuto e non un background sostituto alla creazione delle viste, la scelta varia in base al tipo e target dell'applicazione che stiamo creando.

Ad esempio per un gioco o un'applicazione per bambini la Launch Screen risulterà più appetibile se sarà ricca di animazioni o effetti che esalteranno i contenuti. Contrariamente un'app di consultazione o gestionale potrà presentare una Launch Screen più sobria e funzionale.

Come dalle linee guida di Google o dei principali esperti condivido l'idea di evitare di utilizzare un timer di durata fissa perchè questa non può essere calcolata in modo corretto per i vari fattori presenti quali tempo di caricamento del layout, carico del sistema operativo, tempo di caricamento delle impostazioni iniziali ecc…

Per quanto riguarda invece gli altri 2 metodi solo il programmatore dovrà capire in base alla sua esperienza, al target e tipo dell'applicazione quale sia la scelta migliore.

Spero che questo articolo possa servire, come è successo a me, per chiarire le idee a chiunque si trovi di fronte a dover realizzare una Launch Screen per poter fare la scelta migliore tenedo conto del dominio, dei pro e dei contro.

- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Inglese -> Italiano…
    - Crea un nuovo elenco di parole…
- Copia

- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Inglese -> Italiano…
    - Crea un nuovo elenco di parole…
- Copia

- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Italiano -> Italiano…
    - Crea un nuovo elenco di parole…
- Copia

- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Italiano -> Italiano…
    - Crea un nuovo elenco di parole…
- Copia

- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Spagnolo -> Italiano…
    - Crea un nuovo elenco di parole…
- Copia

- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Italiano -> Italiano…
    - Crea un nuovo elenco di parole…

- Copia
- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Italiano -> Italiano…
    - Crea un nuovo elenco di parole…
- Copia
- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Italiano -> Italiano…
    - Crea un nuovo elenco di parole…
- Copia
- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Italiano -> Italiano…
    - Crea un nuovo elenco di parole…
- Copia
- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Italiano -> Italiano…
    - Crea un nuovo elenco di parole…
- Copia
- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Italiano -> Italiano…
    - Crea un nuovo elenco di parole…
- Copia
- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Italiano -> Italiano…
    - Crea un nuovo elenco di parole…
- Copia
- Aggiungi al Frasario
    - Non ci sono elenchi di parole per Inglese ->

Italiano…
        - Crea un nuovo elenco di parole…
    - Copia
    - Aggiungi al Frasario
        - Non ci sono elenchi di parole per Inglese -> Italiano…
        - Crea un nuovo elenco di parole…
    - Copia
    - Aggiungi al Frasario
        - Non ci sono elenchi di parole per Italiano -> Italiano…
        - Crea un nuovo elenco di parole…
    - Copia
    - Aggiungi al Frasario
        - Non ci sono elenchi di parole per Inglese -> Italiano…
        - Crea un nuovo elenco di parole…
    - Copia

Splash Screen in Android: impariamo a crearle nel modo corretto